PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 11/30, 9/44 — A1

(11) International Publication Number: WO 98/16882

(43) International Publication Date: 23 April 1998 (23.04.98)

(21) International Application Number: PCT/AU97/00678

(22) International Filing Date: 9 October 1997 (09.10.97)

(30) Priority Data:
PO 2920 11 October 1996 (11.10.96) AU

(71) Applicants *(for all designated States except US)*: ERICSSON AUSTRALIA PTY. LTD. [AU/AU]; 61 Riggall Street, Broadmeadows, VIC 3047 (AU). ROYAL MELBOURNE INSTITUTE OF TECHNOLOGY [AU/AU]; 124 LaTrobe Street, Melbourne, VIC 3000 (AU). THE UNIVERSITY OF MELBOURNE [AU/AU]; Grattan Street, Parkville, VIC 3052 (AU).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: WONG, Geoffrey, Peter [AU/AU]; 6/66 Murray Street, Caulfield South, VIC 3162 (AU). O'BRIEN, Fergus [AU/AU]; 9B Moore Street, Coburg, VIC 3058 (AU).

(74) Agent: CARTER SMITH & BEADLE; Qantas House, 2 Railway Parade, Camberwell, VIC 3124 (AU).

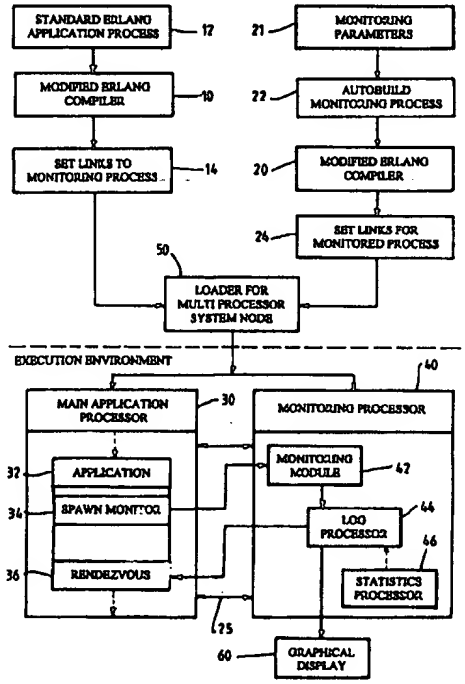
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: A METHOD AND SYSTEM FOR CONTINUOUS SOFTWARE MONITORING

(57) Abstract

A method and system for continuous software monitoring is provided in which the attributes or parameters (21) of the functional computer program to be monitored are specified at the time the functional computer program is compiled and a monitoring computer program is simultaneously compiled by a compiler (20) for parallel execution in a target computer environment (30, 40). The functional and monitoring programs are compiled to include links (14, 24) to enable data to be extracted from the functional computer program, the main application (32), by the monitoring program. The main application (32) is arranged to include a spawn monitor (34) so that each process of the main application which is spawned during execution in the main application processor (30) has its own monitoring process which is spawned in the monitoring processor (40).



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

A METHOD AND SYSTEM FOR CONTINUOUS SOFTWARE MONITORING

This invention relates to a method of and system for continuously monitoring the performance of computer programs and software systems, and particularly throughout the life-cycle of the program or software system.

Computer hardware has become rapidly more sophisticated and powerful since its inception. The software which runs on this hardware has grown in size and complexity at an even more rapid rate. Systems which contain vast amounts of software (such as telecommunications systems) are difficult to comprehend, let alone monitor, accurately access and find faults within.

Established forms of engineering, particularly process engineering, have continuous monitoring processes built in throughout a system. Examples include monitors such as temperature gauges, flow meters and pressure gauges. Because of the rising level of complexity of computer software systems, it is important to be able to use these principles, established in other forms of engineering, and apply them to software engineering.

When a software system is required to meet a certain set of functional requirements it is important to have the software built so that it maximizes its ability to fulfill the requirements. This leads software to be built in a so-called 'optimal' manner which leaves little room for extraneous activities which do not directly contribute to the ability of the software to meet those functional requirements.

It is, however, desirable to monitor functional and non-functional attributes of a software system continuously without the intrusive overhead of inlining the monitoring code. Another important motivation for continuous monitoring is the increasing trend for software development contracts to contain specifications about the non-functional aspects of a system. Of particular interest in these contracts are the non-functional requirements of performance, reliability, usability and

maintainability. It is also desirable to provide a method and system which can continuously monitor attributes of a software system throughout the complete life-cycle of the system at all stages from development, through production to decommission.

According to one aspect of the invention there is provided a method of continuously monitoring the performance or attributes of computer software comprising the steps of:

compiling a functional computer program for execution in a target computer environment;

specifying the attributes of the functional computer program to be monitored at the time the functional computer program is compiled and simultaneously compiling a monitoring computer program for parallel execution in said target computer environment;

said monitoring computer program being adapted to extract data continuously from said functional computer program when said programs are executed in parallel; and

analyzing said data to obtain information about the performance or other attributes of said functional computer program.

The functional and monitoring computer programs may be recorded on any convenient memory or data storage means. According to another aspect of the invention there is provided data storage means including functional computer program and a monitoring computer program for monitoring the performance or other attributes of said functional computer program, said functional computer program being arranged to perform a functional application when executed in a target computer environment, wherein attributes of the functional computer program to be monitored are specified when the functional computer program is compiled, said monitoring computer program being compiled at the same time the functional computer program is compiled, and said monitoring computer program is adapted to extract data continuously from said functional computer program when the

computer programs are executed simultaneously in parallel.

Whilst the functional and monitoring computer programs may be executed in a common processor unit, it is preferred that a separate monitoring processor be provided for executing the monitoring program in parallel to the functional computer program.

This is desirable so that execution and performance of the functional computer program is not adversely affected to a significant extent by execution of the monitoring program and by the processing and analysis of data for monitoring the functional program.

The functional computer program and the monitoring program are preferably compiled so that each process which is spawned in the functional computer program has its own monitoring process which is spawned in the monitoring program. Each process of the functional computer program preferably has its own address space in order to minimize intrusion by the monitoring program. However some intrusive code, automatically supplied during compilation may be necessary to ensure that all monitored information is passed transparently between the functional and monitoring processes.

Preferably, the functional computer program and the monitoring computer program are compiled to include links between the monitoring program and the functional computer program to enable data to be extracted from the functional computer program by the monitoring program. A wide variety of aspects of performance or attributes of functional computer programs may be monitored by the method of the present invention. These include: current execution costs; exceptions in the functional program; the actual time spent in a function; and the use of external resources.

The monitoring process executed by the monitoring program is preferably arranged

to rendezvous with the functional computer program on termination of a functional process performed by the functional computer program. This is particularly desirable if the monitoring process is collecting timing information, e.g. the actual time spent in a function.

The monitoring computer program may also be arranged to terminate a process of the functional computer program upon detection of a fatal error in said process by the monitoring computer program.

According to another aspect of the invention there is provided a system for continuously monitoring the performance or attributes of a functional computer program comprising:

compilation means arranged to compile the functional computer program and a monitoring computer program from a source language into a computer code, the monitoring computer program being arranged to monitor attributes of the functional computer program specified at the time the functional computer program is compiled;

execution means arranged to execute the monitoring computer program simultaneously in parallel with the functional computer program,

wherein the monitoring program is arranged to interact with the functional computer program to extract data continuously from the functional computer program when said programs are executed simultaneously; and

analyzing means for analyzing the data extracted from the functional computer program to continuously monitor the specified attributes of the functional computer program.

The execution means may comprise a common processor unit for executing the functional and monitoring computer programs simultaneously in parallel. However, the execution means preferably includes an application processor for executing the functional computer program (the "main application") and a monitoring processor for simultaneously executing the monitoring program in parallel.

A communications link, preferably a high speed communications bus, is conveniently provided between the main application processor and the monitoring processor to enable data to be extracted from the main application by the monitoring program.

The compilation means for compiling the functional and monitoring computer programs may comprise a single compiler. Alternatively, separate compilers may be used to compile the functional and monitoring computer programs. Where separate processors are used, a multiprocessor loader may be used to load the functional and monitoring programs into the main application processor and the monitoring processor respectively.

When the monitoring processor is loaded with the monitoring program, it preferably includes a monitoring means for receiving and processing data from the main application. The monitoring processor may also include memory means for storing statistical data about the main application which is monitored, and the analyzing means may comprise a log module which is adapted to process the data received from the main application by the monitoring means and statistical data from the memory means to produce information about the performance and attributes of the main application.

The system may conveniently also include display means, preferably one or more graphic displays, for displaying the information about the performance and attributes of the functional computer program produced by the analyzing means.

The system of the present invention is thus able to monitor the performance, quality and attributes of a functional computer program while it is being executed. The system is built around the ability to provide continuous process monitors when compiling the computer code, with the process monitors being written in the same source language. The source language chosen should be able to support multiple processes and process monitoring. There are a large number of suitable languages

available, one of which is Erlang, a source language developed by Telefonaktiebolaget L M Ericsson. Erlang is particularly suitable for use in the present invention because its features include timing support, multiple process support, dynamic module loading, a built in error handling mechanism and real-time garbage collection.

A preferred system in accordance with the invention will now be described, by way of example only, with reference to the accompanying drawing in which:

Figure 1 is a block diagram of a continuous monitoring computer system in accordance with the invention.

Referring to Figure 1, there is shown a continuous monitoring computer system comprising first and second compilers 10 and 20, a main application processor 30, a monitoring processor 40 and a multiprocessor loader 50 for the main application processor 30 and the monitoring processor 40.

Each compiler 10, 20 is a modified Erlang compiler which is arranged to compile a computer program in computer code (C code) from a source code written in Erlang, a programming language developed by Telefonaktiebolaget L.M. Ericsson. It will, however, be appreciated that other programming languages may be used instead of Erlang in the present invention.

The first compiler 10 is arranged to compile a functional computer program, the "main application" from a standard application process 12. The second compiler 20 is arranged to compile a monitoring program from an auto build monitoring process 22 using predetermined monitoring attributes or parameters 21 specified at the time the functional computer program is compiled. The main application and the monitoring program are arranged to be run in parallel simultaneously by the main application processor 30 and by the monitoring processor 40.

The main application processor 30 and the monitoring processor 40 communicate

with each other by a high speed communications link or bus 25 which enables the monitoring processor 40 to extract data continuously from the main application processor 30 as the functional computer program is executed by the main application processor 30. In an alternative embodiment, the main application processor 30 and the monitoring processor 40 may utilize a shared memory to simulate the communications link between the processors.

The main application processor 30 executes the main application 32 which is programmed during compilation by compiler 10 to contain links 14 to the monitoring program. The monitoring program is also programmed during compilation by the compiler 20 to contain links 24 to the main application program to be monitored. The application processor 30 loaded with the main application 32 includes at least one spawn monitor 34 which, when the main application process is being run, passes data about the main application via the high speed bus 25 to the monitoring processor 40.

The monitoring processor 40 loaded with the monitoring program includes a monitoring module 42 for receiving and processing data from the spawn monitor 34 of the main application 32, a log processor 44, and a statistics processor 46 which includes memory means for storing statistical data about the main application process which is monitored.

The log processor 44 of the monitoring processor 40 is adapted to process data about the main application received by the monitoring module 42 and statistical data received from the statistics processor 46. In this manner, the log processor 44 can continuously examine and analyze data extracted from the main application process while it is being run by the main application processor 30 to produce information about the performance and attributes of the functional computer process representing the main application. The log processor 44 is also able to communicate via the high speed bus 25 with a rendezvous section 36 of the main application 32 when a process in the main application is terminated so that the time

taken to run the process in the main application can be monitored, and that any error information can be passed to the main application. If a fatal error in a process of the main application is detected by the monitoring processor 40, the process being monitored can be terminated by the log processor 44 without any change to the system state, and recovery initiated as if the process had not commenced.

The log processor 44 is in communication with one or more display means 60, preferably in the form of a graphical display which can display a graphical analysis about the performance or attributes of the main application process.

The monitoring processor 40 may be programmed to include a wide variety of different continuous process monitors for continuously monitoring different aspects of the performance or attributes of the main application process. Examples of continuous process monitors which may be included in the monitoring processor are:

a real-time profiler which functions as a statistical sampler of a program counter of the main application processor to keep a decaying record of current execution costs;

an exception monitor which monitors exceptions throughout the system (down to the process level);

an actual time monitor which measures the actual time spent in a process (it may also keep a record of other system activity); and

an external resource access monitor 44 which can monitor the use of external resources such as disk channel, of each process.

Various other continuous process monitors may be provided, such as system specific monitors which can present data collected against the system specifications over a variety of time frames enabling the user to examine trends in the system. These include reliability and maintainability monitors.

The manner in which the main application and the monitoring processes are compiled by the compilers 10 and 20 using Erlang as the source code language will now be described.

The system of the present invention for providing continuous process monitors is built around the ability to provide system monitors automatically when compiling code. This provides a system similar to the provision of profiling in existing compilers except that the monitors will be written in the same source language.

In conjunction with choosing this approach it is necessary to select a language with a philosophy which will comfortably support multiple processes and process monitoring. There are a large range of languages available, but Erlang has been chosen as a preferred language for a number of reasons:

- its concurrent programming philosophy;
- the language provision of process support mechanisms;
- its inherent simplicity (based upon a functional language); and
- built in language mechanisms to handle failure.

A primary feature of Erlang is its management of multiple processes as a fundamental part of the system. The feature is critical to the construction of a monitoring Erlang compiler and to plan for this capability in the construction phase .

The monitor is designed for a multiple processor environment although in a prototype system Unix process support (sockets and Unix communications between machines: internet sockets) is utilized to support multiple processes. To do this each process is linked into an Erlang runtime library which provides this communication support. Processes communicate via sockets in (or internet ports) with their Unix pid as the socket identifier. This (with the machine id) is the identifier that gets created when a process is spawned. The Erlang runtime library also contains other support code for Erlang internals such as list and tuple

management. The actual compilation takes place in a number of phases:

1. Lexing/parsing - this phase will produce an abstract syntax tree.
2. Optimization - break the tree into basic blocks for local expression optimization. Try 'global' (within a module) optimization; also do (importantly) last call optimization.
3. Code production - this is essentially the transformation of the tree into C code.
4. The C compiler then performs other optimizations.

Since the target language is Erlang the monitoring processes are also written in Erlang. The compiler includes code in the functional process to spawn the hand coded monitoring process automatically using the existing Erlang system for process creation and control. Monitors are intended to be processes that simply collect information. These monitoring processes can then pass data along to other processes responsible for the analysis and presentation of that data. Since monitors are written in the source language (Erlang) the process communication mechanisms in Erlang can be exploited to forward this information to other processes which are responsible for displaying and analyzing the collected information. The compiler which compiles from Erlang to C-code can support the following features: dynamic loading, run-time garbage collection and multiple process support (over multiple machines).

Depending on what information the monitoring process is collecting it may be necessary for the monitoring process and the functional process to rendezvous on the termination of the functional process. This is particularly the case if the monitoring process is collecting timing information. This rendezvous can create a performance overhead in cases where the execution of the monitoring process is held up for some reason. It is intended that the monitoring processes be small and straight forward processes which contain little functionality other than data gathering mechanisms. By virtue of the monitoring process being simple and small it is predicted that these processes will generally be ready to rendezvous with the

functional process when the functional process is ready to terminate.

The present invention therefore provides an effective method and system for continuously monitoring the performance or other attributes of a functional computer program which does not adversely affect the execution of the functional computer program to a significant extent.

In the method and system of the present invention, specifying the attributes to be monitored and compiling the monitoring program at the time the functional computer program is compiled allows the specified attributes to be monitored throughout the life-cycle of the functional computer program.

It will be appreciated that various modifications may be made to the embodiment described above without departing from the scope and spirit of the present invention. For instance, whilst the present invention has been described above with reference to Erlang as the source language, other source languages having similar properties and functions may be used.

CLAIMS

1. A method of continuously monitoring the performance or other attributes of computer software comprising the steps of:

compiling a functional computer program for execution in a target computer environment;

specifying the attributes of the functional computer program to be monitored at the time the functional computer program is compiled and simultaneously compiling a monitoring computer program to monitor said attributes for parallel execution in said target computer environment;

said monitoring program being adapted to extract data continuously from said functional computer program when said programs are executed in parallel; and

analyzing said data to obtain information about said specified attributes of said functional computer program.

2. A method according to claim 1, wherein the functional computer program and the monitoring computer program are compiled to include links between the monitoring program and the functional computer program to enable data to be extracted from the functional computer program by the monitoring program.

3. A method according to claim 1 or claim 2, wherein the functional computer program and the monitoring program are compiled so that each functional process which is spawned in the functional computer program has its own monitoring process which is spawned in the monitoring program.

4. A method according to claim 2 or claim 3, wherein a monitoring process executed by the monitoring program is arranged to rendezvous with the functional computer program on termination of a functional process performed by the functional computer program.

5. A method according to any one of claims 1 to 4, wherein the

attributes which are monitored by the monitoring program include any one or more of the following: current execution costs; exceptions in the functional program; the actual time spent in a function; the use of external resources; reliability; or maintainability.

6. A method according to any one of the preceding claims wherein the functional and monitoring computer programs are written in the same source language.

7. A method according to claim 6 wherein the source language is Erlang.

8. A method according to claim 6 or claim 7, wherein the functional and monitoring computer programs are compiled from the source language into computer code by a single compiler.

9. A method according to claim 6 or claim 7, wherein the functional and monitoring computer programs are compiled from the source language by separate compilers.

10. A method according to any one of the preceding claims wherein the functional computer program and the monitoring computer program are executed in parallel in a common processor unit.

11. A method according to any one of claims 1 to 9, wherein the functional computer program and the monitoring computer program are executed in separate processors connected by a high speed communications link.

12. A method according to any one of the preceding claims, wherein the monitoring computer program is arranged to terminate a process of the functional computer program upon detection of a fatal error in said process by the monitoring computer program.

13. Data storage means including a functional computer program and a monitoring computer program for monitoring the performance or other attributes of said functional computer program, said functional computer program being arranged to perform a functional application when executed in a target computer

environment, wherein attributes of the functional computer program to be monitored are specified when the functional computer program is compiled, said monitoring computer program being compiled at the same time the functional computer program is compiled, and said monitoring computer program is adapted to extract data continuously from said functional computer program when the computer programs are executed simultaneously in parallel.

14. Data storage means according to claim 13, wherein the functional computer program and the monitoring computer program include links between the monitoring program and the functional computer program to enable data to be extracted from the functional computer program by the monitoring program.

15. Data storage means according to claim 14, wherein the functional computer program and the monitoring program are arranged so that each functional process which is spawned in the functional computer program has its own monitoring process which is spawned in the monitoring program.

16. Data storage means according to claim 14 or claim 15, wherein a monitoring process executed by the monitoring program is arranged to rendezvous with the functional computer program on termination of a functional process performed by the functional computer program.

17. Data storage means according to any one of claims 14 to 16 wherein the monitoring program is arranged to terminate a process of the functional computer program upon detection of a fatal error in said process by the monitoring computer program.

18. A system for continuously monitoring the performance or other attributes of a functional computer program comprising:

means arranged to compile a functional computer program and a monitoring computer program from source language into a computer code; the monitoring computer program being arranged to monitor attributes of the functional computer program specified at the time the functional computer program is compiled;

execution means arranged to execute the monitoring computer program simultaneously in parallel with the functional computer program;

wherein the monitoring program is arranged to interact with the functional computer program to extract data continuously from the functional computer program when said programs are executed simultaneously; and

analyzing means for analyzing the data extracted from the functional computer program to continuously monitor the specified attributes of the functional computer program.

19. A system according to claim 18, wherein the execution means comprises a common processing unit including an application processor for executing the functional computer program and a monitoring processor for executing the monitoring computer program simultaneously with the functional computer program.

20. A system according to claim 18, wherein the execution means comprises an application processor for executing the functional computer program and a separate monitoring processor for simultaneously executing the monitoring program in parallel.

21. A system according to claim 20, wherein the application processor and the monitoring processor are connected by a high speed communications link.

22. A system according to any one of claims 19 to 21, wherein the application processor includes a spawn monitor which, when a process of the functional computer program is spawned in the application processor, communicates with the monitoring processor to pass data about the functional program to the monitoring processor.

23. A system according to claim 22, wherein the monitoring processor when loaded with the monitoring program includes monitoring means for receiving and processing data from the spawn monitor, memory means for storing statistical

data about the functional computer program and a log processor for analyzing said data from the spawn monitor and said statistical data to produce information about the specified attributes of the functional computer program.

24. A system according to any one of claims 19 to 23, wherein the monitoring processor is arranged to communicate with the functional computer program on termination of a process in the application processor.

25. A system according to any one of claims 19 to 23, wherein the monitoring processor is arranged to communicate with the application processor upon detection of a fatal error in a process of the functional computer program to terminate said process.

26. A system according to any one of claims 19 to 25, wherein the monitoring processor includes a real-time profiler which continuously monitors the current execution cost of the functional computer program.

27. A system according to any one of claims 19 to 26, wherein the monitoring processor includes an exception monitor which monitors exceptions in the functional computer program.

28. A system according to any one of claims 19 to 27, wherein the monitoring processor includes an actual time monitor which measures the actual time spent in a process of the functional computer program.

29. A system according to any one of claims 19 to 28 wherein the monitoring processor includes an external resource access monitor which monitors the use of external resources used by the functional computer program.

30. A system according to claim 23 wherein the monitoring processor includes a statistical monitor for analyzing statistical data from the memory means over a number of time frames.

31. A system according to any one of claims 18 to 30 wherein the

compilation means comprises a single compiler.

32. A system according to any one of claims 19 to 30 wherein the compilation means comprises separate compilers for compiling the functional computer program and the monitoring computer program respectively.

33. A system according to claim 32, including a multi-processor loader for loading the functional and monitoring computer programs into the application processor and the monitoring processor respectively.

34. A system according to any one of claims 31 to 33 wherein the or each compiler comprises an Erlang compiler which is arranged to compile the functional and monitoring computer programs written in Erlang into computer code.

35. A system according to any one of claims 18 to 34, wherein an autobuild monitoring process is provided to construct the monitoring computer program from specified monitoring parameters.

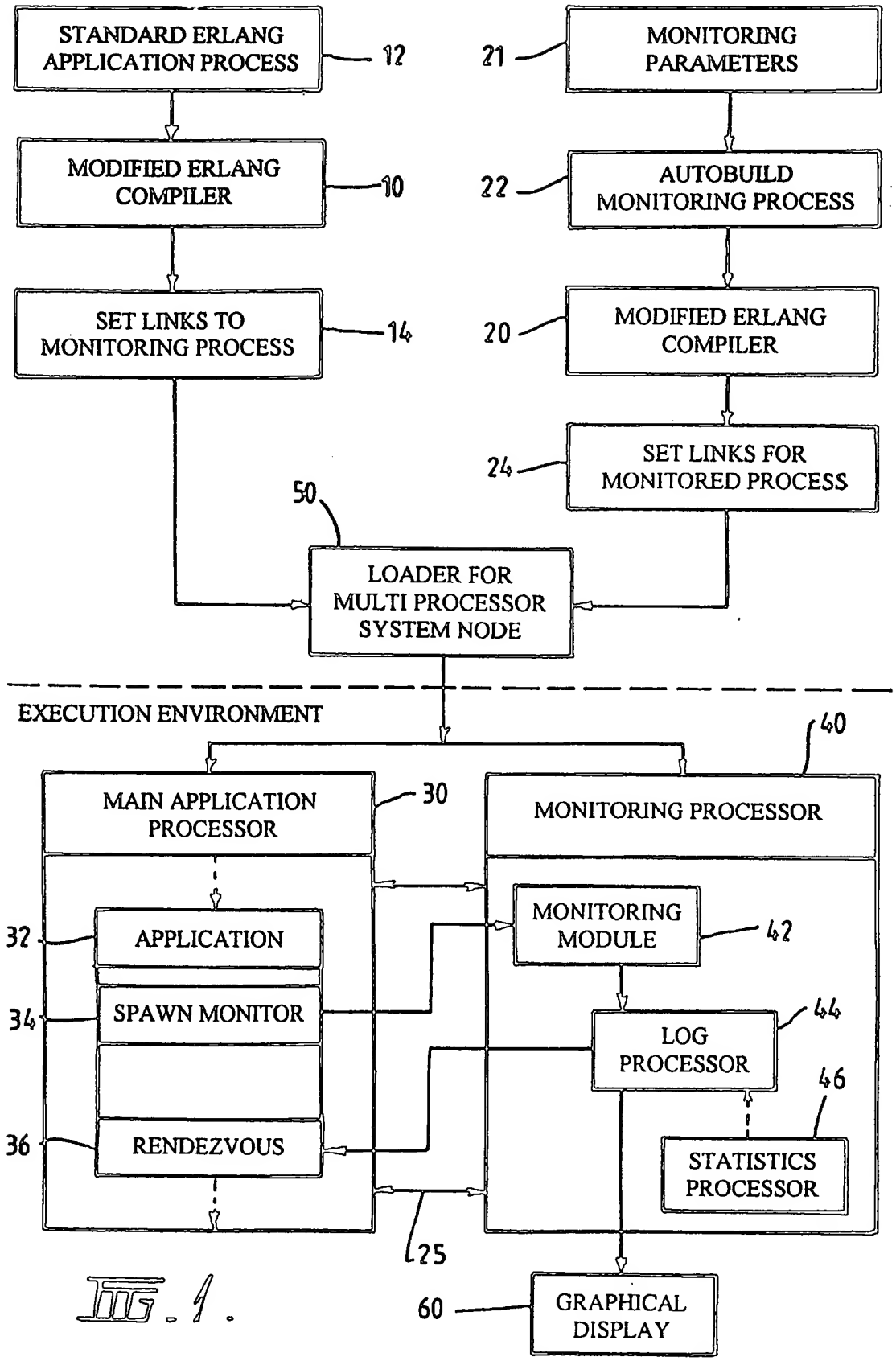

STANDARD ERLANG APPLICATION PROCESS
12
MODIFIED ERLANG COMPILER
10
SET LINKS TO MONITORING PROCESS
14
21
MONITORING PARAMETERS
22
AUTOBUILD MONITORING PROCESS
20
MODIFIED ERLANG COMPILER
24
SET LINKS FOR MONITORED PROCESS
50
LOADER FOR MULTI PROCESSOR SYSTEM NODE
EXECUTION ENVIRONMENT
40
MAIN APPLICATION PROCESSOR
30
MONITORING PROCESSOR
32
APPLICATION
34
SPAWN MONITOR
36
RENDEZVOUS
MONITORING MODULE
42
LOG PROCESSOR
44
46
STATISTICS PROCESSOR
25
60
GRAPHICAL DISPLAY

FIG. 1.

INTERNATIONAL SEARCH REPORT

International Application No.
PCT/AU 97/00678

A. CLASSIFICATION OF SUBJECT MATTER

Int Cl[6]: G06F 11/30, 9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC as above (G06F + keywords)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
AU: IPC (G06F 11/30, 32, 34, G06F 9/44, 46, 455)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WPAT, IEEE COMPUTER (monitor:, progr:, contin:, software)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	EP 769 745 A1 (Sun Microsystems) 23 April 1997 Abstract, figures, summary	1, 4
X Y	US 5 528 753 (IBM) 18 June 1996 whole document	1, 5, 18 2, 4, 6, 26
	"Parallel Computing in Networks of workstations with Paralex" (Renzo et al.) IEEE Transactions on Parallel and Distributed Systems, Vol. 7, No. 4, April 1996, pp 371-384	
X	whole document, esp. pp. 1-5, 7-13	1, 2, 6, 8-10, 11, 13, 14, 18

[X] Further documents are listed in the continuation of Box C

[X] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 October 1997	03 NOV 1997
Name and mailing address of the ISA/AU AUSTRALIAN INDUSTRIAL PROPERTY ORGANISATION PO BOX 200 WODEN ACT 2606 AUSTRALIA Facsimile No.: (02) 6285 3929	Authorized officer Dale Siver Telephone No.: (02) 6283 2196

INTERNATIONAL SEARCH REPORT

International Application No.
PCT/AU 97/00678

C (Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	"On-Line Monitoring: A tutorial (Schroeder) IEEE Computer Vol. 28,#6, June 1996, pp. 72-78	
X Y	whole document	1, 5, 6 2, 4, 12, 17, 24-26
	EP 636 985 A (IBM) 1 February 1995	
X	abstract, figures	1, 6, 8, 18
X	AU 53775/90 A (Sun Microsystems) 28 March 1991	1, 5, 6, 8, 18, 20-24, 26-28
Y	WO 95/25304 (Green Hills Software) 21 September 1995	1, 2, 5, 6
Y	US 5 313 616 (88 Open Consortium) 17 May 1994 abstract, figures, background	1, 2, 5, 6, 18

INTERNATIONAL SEARCH REPORT
Information on patent family members

International Application No.
PCT/AU 97/00673

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
EP	769745	JP	9160859				
US	5528753						
EP	636985	JP	7093232	US	5636376		
AU	53775/90	GB	2236202	HK	537/94	SG	331/94
WO	9525304	AU	19946/95	CA	2185222	EP	750767
US	5313616						

END OF ANNEX